

Efficient Symbol Sorting for High Intermediate Recovery Rate of LT Codes

Ali Talari, Behzad Shahrabi, and Nazanin Rahnavard

Oklahoma State University, Stillwater, OK 74078

Emails: {ali.talari, behzad.shahrabi, nazanin.rahnava} @okstate.edu

Abstract

LT codes are modern and efficient *rateless forward error correction* (FEC) codes with close to channel capacity performance. Nevertheless, in *intermediate range* where the number of received encoded symbols is less than the number of source symbols, LT codes have very low recovery rates.

In this paper, we propose a novel algorithm which significantly increases the intermediate recovery rate of LT codes, while it preserves the codes' close to channel capacity performance. To increase the intermediate recovery rate, our proposed algorithm *rearranges* the transmission order of the encoded symbols exploiting their structure, their transmission history, and an estimate of the channel's erasure rate. We implement our algorithm for conventional LT codes, and numerically evaluate its performance.

I. INTRODUCTION

LT codes are modern, efficient, and flexible *rateless forward error correction* (FEC) codes [1]. An LT encoder at a source S can potentially generate a limitless number of encoded symbols $c_i, i \in \{0, 1, \dots\}$ from k source symbols $\underline{x} = \{x_1, x_2, \dots, x_k\}$. The decoder at a destination D can successfully recover \underline{x} from any $k\gamma$ of received encoded symbols, where γ is the received *coding overhead*. Let γ_{succ} denote the required overhead for a successful decoding. Conventional LT codes can achieve $z \approx 1$ with high probability at γ_{succ} 's slightly larger than one, where z denotes the ratio of the number of recovered source symbols to k at D . Note that, for LT codes, γ_{succ} is fixed and known [1].

Each LT code is identified by a *degree distribution*. In LT encoding, first an encoded symbol degree d is chosen from a degree distribution, $\{\Omega_1, \Omega_2, \dots, \Omega_k\}$, where Ω_i is the probability that $d = i$. This degree distribution is also shown by its generator polynomial $\Omega(y) = \sum_{i=1}^k \Omega_i y^i$. Next, d x_j 's are chosen uniformly at random from \underline{x} , and are *XORed* to generate an encoded symbol c_i . Let $\varepsilon \in [0, 1)$ denote the channel erasure rate. Therefore, S needs to generate on average $m = \frac{k\gamma_{succ}}{1-\varepsilon}$ encoded symbols so that D can collect $k\gamma_{succ}$ of them for a successful decoding.

In LT decoding at D , if a newly delivered $c_i, i \in \{1, 2, \dots, m\}$ has degree-one, it can decode one $x_j, j \in \{1, 2, \dots, k\}$. If the degree of c_i is larger than one then previously recovered x_j 's, if any, are removed from c_i to decrease its degree. If the degree of c_i is reduced to one, this symbol is similar to a new degree-one c_i . On the other hand, if the remaining degree is larger than one, the symbol is *buffered*. If any new x_j is recovered in the previous step, it is removed from the buffered c_i 's. Hence, some buffered c_i 's may acquire a degree equal to one, resulting in recovery of further x_j 's. This procedure is repeated iteratively until no more degree-one c_i emerges.

Intermediate range of LT codes refers to the case where the transmission is still in progress, i.e., $0 \leq \gamma < 1$ [2]. Since conventional LT codes [1] are designed to have an almost complete recovery of \underline{x} ($z \approx 1$) for certain γ_{succ} 's > 1 , they have low intermediate recovery rates ($z \approx 0$) at $0 \leq \gamma < 1$. However, in many applications where *partial* recovery of the source symbols from the incomplete received encoded symbols is still beneficial, intermediate recovery rate becomes important. For instance, in multimedia transmission the receiver can play a lower quality of the multimedia contents from the incomplete recovered data. This motivates the design of an algorithm to improve the intermediate recovery rate of existing LT codes.

In order to obtain a high z (as close as possible to γ) in $0 \leq \gamma < 1$, each delivered c_i should decode (on average) one x_j instead of being buffered. We propose an algorithm to *rearrange* the transmission order of c_i 's based on their transmission history, structure of each c_i , and an estimate of ε , such that each delivered c_i can decode one x_j with high probability. By employing our proposed algorithm, the intermediate performance of a given LT code improves significantly, while the recovery rate of \underline{x} at γ_{succ} remains intact. In other words, the code remains capacity achieving similar to its original setup.

This paper is organized as follows. Section II reviews the exiting work on intermediate performance of LT codes. In Section III, we propose our novel sorting algorithm. Next, Section IV reports the performance evaluation of our proposed algorithm. Finally, Section V concludes the paper.

II. RELATED WORK

In [2], Sanghavi has shown that $0 \leq \gamma < 1$ and equivalently $0 \leq z < 1$ can be divided into three regions of $z \in [0, \frac{1}{2}]$, $z \in [\frac{1}{2}, \frac{2}{3}]$, and $z \in (\frac{2}{3}, 1)$. In each region, the upper bound on z for all rateless codes versus γ is formulated, and the optimum $\Omega(y)$'s to gain these upper bounds are provided. The $\Omega(y)$'s provided for each region perform optimally in that specific region only, thus they have a low z compared to γ in other two regions. Further, provided $\Omega(y)$'s are not capacity achieving, and they are designed for asymptotic case (infinite k) and may not be employed in practice where k is finite.

Authors in [3] have proposed *Growth codes*, which are designed to increase the number of recovered x_j 's in intermediate range in wireless sensor networks. In Growth coding, S gradually increases the degree of c_i 's on-the-fly based on the value of z (which is known to S by feedbacks received from D), such that the instantaneous decoding probability of each delivered c_i is maximized. Growth codes only consider the instantaneous recovery probability of each c_i , thus they do not have a close to capacity performance. More importantly, Growth codes require a lot of feedbacks from the receiver.

Authors in [4] have proposed to employ multiple feedbacks to transmit c_i 's in the order of their *degree* to increase the intermediate recovery rate. Since the decoding of c_i 's with lower degrees depends on the recovery of a smaller subset of x_j 's, they have a higher probability of decoding an x_j in D at the beginning of transmission. Consequently, an improvement is observed in the intermediate recovery rate of LT codes. However, the algorithm proposed in [4] cannot outperform the code of [3], [2] in intermediate range. Besides, we show that our proposed algorithm always surpasses the algorithm in [4].

In our recent work [5], we have designed several $\Omega(y)$'s for LT codes to obtain optimum intermediate performance *throughout* the intermediate range rather than a single γ [2]. The LT codes designed in [5] do not require channel information or feedbacks in contrast to [3], [4]. However, similar to [2] the codes designed in [5] cannot achieve channel capacity.

III. THE PROPOSED ALGORITHM

A. Discussion and Idea

In the previous section, we observed a *trade-off* between being channel capacity achieving and having a high intermediate recovery rate. Some LT codes have optimum intermediate performance but they cannot achieve channel capacity [5], [2]. On the other hand, the algorithms that are proposed to improve the intermediate recovery rate of LT codes with close to channel capacity such as [4] cannot outperform codes of the first group in intermediate range.

The reason for this trade-off is that the codes in the first group have $\Omega(y)$'s that result in generation of a large fraction of low-degree c_i 's. Low-degree c_i 's have a higher probability of decoding a source symbol when z is small. However as z grows, some of the received c_i 's become redundant and cannot recover any x_j due to earlier recovery of all their adjacent x_j 's. Therefore, these codes cannot have a close to channel capacity performance.

On the other hand, $\Omega(y)$'s of codes with close to channel capacity performance result in generation of c_i 's with much higher degrees. Decoding of high-degree c_i 's depends on the recovery of many x_i 's at D , thus in intermediate range these c_i 's are mostly buffered for a later decoding. Therefore, for these codes

z does not grow considerably with γ . However, the buffered high-degree c_i 's are simultaneously decoded together and recover \underline{x} at γ_{succ} , which makes the code capacity achieving.

We can see that in LT codes with capacity achieving performance, each c_i eventually decodes close to one x_j on average, since all k x_j 's are decoded from $k\gamma_{succ}$ c_i 's, which are slightly more than k symbols. Consequently, if c_i 's are transmitted in the *order* that they are *decoded*, we can significantly improve the intermediate recovery rate of *capacity achieving* LT codes. We propose our algorithm to transmit c_i 's in this *correct order* for two cases of constant and varying ε .

B. Algorithm for Constant ε

Similar to [6], we assume that an estimate of the channel erasure rate, ε , is available at S . Since we assume that D generates no feedbacks, our algorithm is designed to be implemented on the encoder side. Therefore, our algorithm can exploit the information available in S only, and the decoder remains intact.

In conventional LT coded symbol transmission, S generates $m = \frac{k\gamma_{succ}}{1-\varepsilon}$ random c_i 's from a capacity achieving degree distribution $\Omega(\cdot)$ such as *Robust-Soliton* degree distribution [1]. After transmission, D receives $k\gamma_{succ}$ encoded symbols, which results in a successful decoding of \underline{x} . This method has a poor intermediate recovery rate as we later show.

In our proposed scheme, after generating m c_i 's, S performs as follows. S maintains a probability vector $\underline{\rho} = [\rho(1), \rho(2), \dots, \rho(k)]$, in which $\rho(j)$ represents the probability that x_j is still *not recovered* at D . Clearly, S sets $\underline{\rho}$ to an all-one vector when the transmission has not started yet since no x_j is recovered at D . At each transmission and based on $\underline{\rho}$, S finds a c_i that has the highest probability of recovering an x_i at D (as we later describe in Algorithm 1). Next, S transmits c_i and updates $\rho(j), j \in \mathcal{N}(c_i)$, where $\mathcal{N}(c_i)$ represents the set of indices of x_i 's *XORed* together to generate c_i . S continues until all m c_i 's are transmitted.

An encoded symbol c_i with degree d , i.e., $|\mathcal{N}(c_i)| = d$, where $|\cdot|$ represents the cardinality of a set, can recover a source symbol x_j iff all $x_k, k \in \{\mathcal{N}(c_i) - j\}$ has already been recovered. Let $\underline{p}_{dec} = \{p_{dec}(1), p_{dec}(2), \dots, p_{dec}(m)\}$, where $p_{dec}(i)$ is the probability that c_i can recover a source symbol at D , and $p_{dec}(i) = 0$ if c_i has been previously transmitted. Since at the beginning of transmission no source symbol is still recovered, we have $p_{dec}(i) = 0$ if $|\mathcal{N}(c_i)| > 1$, i.e., c_i 's with degrees larger than one cannot decode any x_j at D . Besides, for $|\mathcal{N}(c_i)| = 1$ we have $p_{dec}(i) = (1 - \varepsilon)$, i.e. only degree-one encoded symbols that are not erased by the channel (with probability $(1 - \varepsilon)$) can recover a source symbol.

We can see that at the beginning of transmission degree-one c_i 's have the highest probabilities of decoding an x_j at D . Therefore, S transmits degree-one c_i 's with $\mathcal{N}(c_i) = \{j\}$, and updates $\rho(j) = \varepsilon\rho_{old}(j)$, where $\rho_{old}(j)$ is the value of $\rho(j)$ before c_i was transmitted.

Next, we consider a degree-two $c_i, \mathcal{N}(c_i) = \{j, k\}$. c_i can recover x_j with probability $(1 - \varepsilon)(1 - \rho(j))\rho(k)$, which is the probability that c_i is not dropped on channel, x_j has not been recovered previously, and the x_k has already been recovered. Similarly, c_i can recover x_k with probability $(1 - \varepsilon)(1 - \rho(k))\rho(j)$. Consequently, $p_{dec}(i) = (1 - \varepsilon)[(1 - \rho(k))\rho(j) + (1 - \rho(j))\rho(k)]$. Assume $\forall l \neq i, p_{dec}(i) > p_{dec}(l)$, i.e. c_i has the highest probability of decoding an x_j at D . Therefore, S transmits c_i next and sets $\rho(j) = \rho_{old}(j)(1 - (1 - \varepsilon)(1 - \rho_{old}(k)))$ and $\rho(k) = \rho_{old}(k)(1 - (1 - \varepsilon)(1 - \rho_{old}(j)))$.

Further, we consider $c_i, |\mathcal{N}(c_i)| = d$. If c_i is successfully delivered to D , it can possibly decode $x_j, j \in \mathcal{N}(c_i)$. Similar to low degree c_i 's, x_j can be decoded with probability $(1 - \varepsilon)\rho(j) \prod_{v \in \mathcal{N}(c_i), v \neq j} (1 - \rho(v))$.

Therefore, $p_{dec}(i) = (1 - \varepsilon) \sum_{l \in \mathcal{N}(c_i)} [\rho(l) \prod_{v \in \mathcal{N}(c_i), v \neq l} (1 - \rho(v))]$. If $p_{dec}(i) > p_{dec}(l), \forall l \neq i$, S transmits c_i and updates $\rho(j) = \rho_{old}(j)[1 - (1 - \varepsilon) \prod_{v \in \mathcal{N}(c_i), v \neq j} (1 - \rho_{old}(v))], j \in \mathcal{N}(c_i)$.

We summarize our proposed sorting scheme in Algorithm 1. The output of Algorithm 1 is the suitable rearranged transmission order $\underline{\pi}$ of c_i 's that substantially improves z in $0 \leq \gamma < 1$. In this algorithm, $\text{argmax}(\underline{p}_{dec})$ is a function that returns i where $\forall j \neq i, p_{dec}(i) > p_{dec}(j)$. Further, if c_i and c_l both have the highest probability of decoding of an x_j , i.e., $p_{dec}(l) = p_{dec}(i)$, then $\text{argmax}(\underline{p}_{dec})$ returns the index

of c_i or c_l , whichever has the *lowest* degree. Further, if c_i and c_l have equal degrees (similar to degree one c_i 's at the beginning of transmission), $\text{argmax}(p_{dec})$ randomly returns one of the indices. Clearly, we are assigning earlier transmission priority to lower degree symbols, which we later show is important for $\varepsilon \rightarrow 1$.

Algorithm 1 The proposed symbol sorting algorithm

```

Initialize:  $\underline{\pi} = \emptyset$ ,  $\underline{\rho} = \{1\}_{1 \times k}$ 
while  $|\underline{\pi}| < m$  do
  for  $j = 1$  to  $m$ ,  $j \notin \underline{\pi}$  do
     $p_{dec}(j) = (1 - \varepsilon) \sum_{l \in \mathcal{N}(c_i)} [\rho(l) \prod_{v \in \mathcal{N}(c_i), v \neq l} (1 - \rho(v))]$ 
  end for
   $i^* = \text{argmax}(p_{dec})$ 
   $\underline{\pi} = [\underline{\pi}, i^*]$ 
  for  $j \in \mathcal{N}(c_{i^*})$  do
     $\rho(j) = \rho_{old}(j)[1 - (1 - \varepsilon) \prod_{v \in \mathcal{N}(c_{i^*}), v \neq j} (1 - \rho_{old}(v))]$ 
  end for
end while

```

Our proposed algorithm increases the coding complexity of LT codes from $O(k \log k)$ [1] to $O(k^2)$, while it does not deteriorate the decoding complexity.

Further, in our proposed algorithm all c_i 's need to be generated and sorted before the transmission can start in contrast to the conventional LT coding where each c_i can be independently transmitted upon generation. Therefore, some delays may be introduced.

However, this delay can be easily eliminated with the following procedure. Clearly, the performance of our proposed scheme is independent of \underline{x} 's contents and only depends on $\mathcal{N}(c_i), i \in \{1, 2, \dots, m\}$ and $\underline{\pi}$. Therefore, before the transmission starts, S generates c_i 's from a dummy \underline{x} , and obtains an off-line version of $\underline{\pi}_{\text{off-line}}$. S saves $\mathcal{N}_{\text{off-line}}(c_i)$, and $\underline{\pi}_{\text{off-line}}$ for a later use. When the actual encoding starts, \underline{x} of interest replaces the dummy \underline{x} , and S generates m c_i 's *in the order* dictated by $\underline{\pi}_{\text{off-line}}$, XORing $x_j, j \in \mathcal{N}_{\text{off-line}}(c_i)$. In this way, each c_i can be transmitted upon generation and the delay is completely eliminated at the cost of some data storage.

In the next section, we extend Algorithm 1 to the case where ε varies.

C. Algorithm for Varying ε

Assume that S has generated m c_i 's considering ε . Assume that the erasure rate of the channel changes to ε_{new} when $\frac{k\gamma_c}{1-\varepsilon}$ symbols has already been transmitted so that $\frac{k(\gamma_{succ}-\gamma_c)}{1-\varepsilon}$ c_i 's are still remaining to be transmitted.

If $\varepsilon_{new} > \varepsilon$, less than $k\gamma_{succ}$ c_i 's would be collected by D , making the full decoding impossible. In this case, S generates $t = (\frac{1}{1-\varepsilon_{new}} - \frac{1}{1-\varepsilon})k(\gamma_{succ} - \gamma_c)$ new c_i 's, and adds them to the queue of c_i 's to be transmitted to ensure the delivery of $k\gamma_{succ}$ c_i 's to D . Next, S rearranges all c_i 's in the queue according to Algorithm 1 and continues the transmission.

In the second case for $\varepsilon_{new} < \varepsilon$, S randomly drops $1 - \frac{1-\varepsilon}{1-\varepsilon_{new}}$ fraction of remaining c_i 's from the transmission queue. This limits the number of delivered c_i 's to $k\gamma_{succ}$, hence the code maintains its close to channel capacity performance.

If the erasure rate of the channel varies several times, the same procedures are followed after each change. We assume that ε_{new} is known to S , which can be realized by receiving few feedbacks from the receiver.

Note that the symbol dropping procedure described above is similar to *puncturing LDPC* [7] and *turbo* codes [8] to achieve a certain higher coding rate for these codes.

IV. EVALUATION OF THE PROPOSED ALGORITHM

We emphasize that our proposed algorithm can be applied to an LT code with any degree distribution to increase its intermediate recovery rate when ε is available at S . The advantage of this algorithm is that if the code is capacity achieving, it remains capacity achieving after employing our algorithm.

To evaluate the performance of our proposed algorithm, we implement it for two well-known LT codes. The first code we employ is the LT code used in *Raptor codes* [9] with degree distribution $\Omega(y)$ given below and $k = 1000$. Since low error floors cannot be achieved in intermediate range, the precoding phase of Raptor codes can be skipped.

$$\begin{aligned} \Omega(y) = & 0.00797x + 0.49357x^2 + 0.16622x^3 + 0.07265x^4 + 0.08256x^5 \\ & + 0.05606x^8 + 0.03723x^9 + 0.05559x^{19} + 0.02502x^{65} + 0.00314x^{66}. \end{aligned}$$

The second code is an LT code with $k = 1000$ and Robust-Soliton degree distribution [1] with parameters $c = 0.05$ and $\delta = 0.01$.

Figures 1 and 2 illustrate the improvement made in z for aforementioned Raptor and LT codes employing our proposed sorting algorithm.

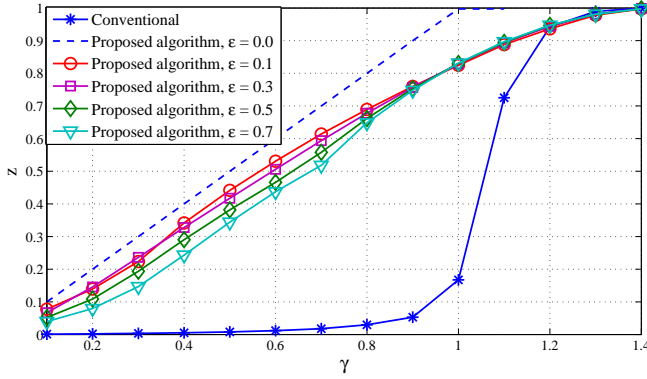


Fig. 1. Improvement made in the intermediate recovery rate of a Raptor code proposed in [9] employing proposed algorithms for various ε 's versus γ .

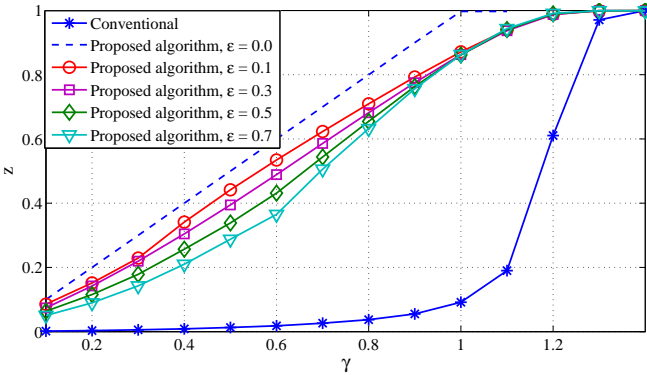


Fig. 2. Improvement made in the intermediate recovery rate of an LT code [1] employing proposed algorithms for various ε 's versus γ .

From Figures 1 and 2, we can see a significant improvement in z for both codes. For instance at $\gamma = 1$, for the first and the second code we can see 382% and 843% improvement in z , respectively. We can observe that the amount improvement depends on the value of ε . The rationale behind this is that when ε decreases our algorithm can estimate the recovery probability of x_j 's more accurately, which results in a more efficient reordering of c_i 's. As ε becomes larger, the ordering of c_i 's becomes less accurate. It is worth noting that the conventional transmission of LT codes results in the same curve of z regardless of ε 's value.

A. Upper and Lower Bounds on Algorithm's Performance

As $\varepsilon \rightarrow 1$, S cannot make a meaningful estimation about the recovery of x_j 's at D , and $\underline{\rho}$ always remains an all-one vector. Since in our proposed algorithm, c_i 's with lower degrees have higher priority of transmission, for $\varepsilon \rightarrow 1$ our algorithm approaches to the case where c_i 's are only sorted based on their degrees in $\underline{\pi}$. Consequently, in this case our proposed algorithm boils down to the algorithm proposed in [4]. As a result, the improvement made by our algorithm is lower bounded by the results of [4]. Moreover, for $\varepsilon \rightarrow 0$, S can estimate which x_j 's are being decoded with a high accuracy, thus more exact $\underline{\pi}$ can be acquired, and the intermediate performance approaches the ideal upper bound, i.e., $z = \gamma$.

The upper and the lower bounds on our proposed scheme are depicted in Figures 3 and 4 for distribution of Raptor and LT codes, respectively. For comparison, we have also provided the recovery rate curves of Growth codes [3] and conventional LT transmission for the same codes. Further, the lower bound illustrates the performance of the scheme proposed in [4].

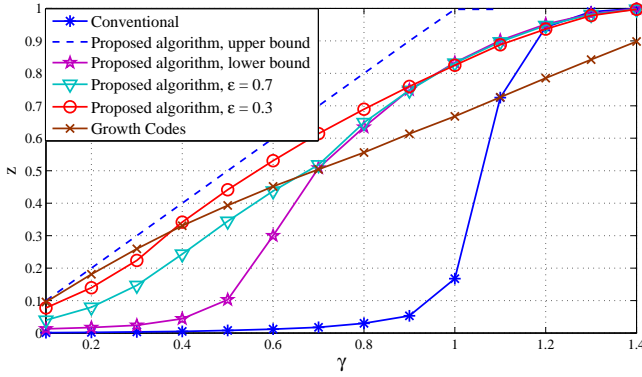


Fig. 3. The upper and the lower bounds on the improvement made by our proposed scheme for Raptor codes [9], compared to the intermediate recovery rate of Growth codes and conventional LT codes.

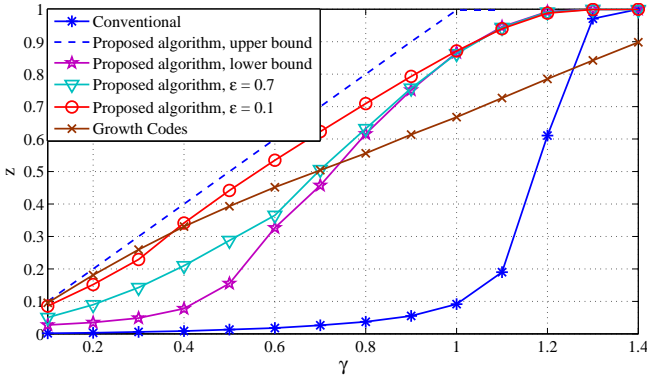


Fig. 4. The upper and the lower bounds on the improvement made by our proposed scheme for LT codes [1], compared to the intermediate recovery rate of Growth codes and conventional LT codes.

Figures 3 and 4 show considerable improvement for intermediate recovery rate of both employed LT codes for any value of ε . We can see that Growth codes outperform our propose algorithm only for a small region of γ , while they cannot have a close to channel capacity performance.

B. Algorithm's Performance for Varying ε

As described earlier, if ε increases, S needs to generate some new c_i 's. Assume that S is transmitting m Raptor encoded c_i 's (without precoding) generated for $\varepsilon = 0.3$. Also assume that ε increases to $\varepsilon_{new} = 0.5$ at $\gamma_c = 0.5$. Based on our proposed algorithm, S adds $t = \lceil 0.5714k(\gamma_{succ} - \gamma_c) \rceil$ new c_i 's to the queue,

and updates $\underline{\pi}$ accordingly. We have depicted z versus γ for this case in Figure 5. For comparison, we have also depicted z for constant $\varepsilon \in \{0.3, 0.5\}$.

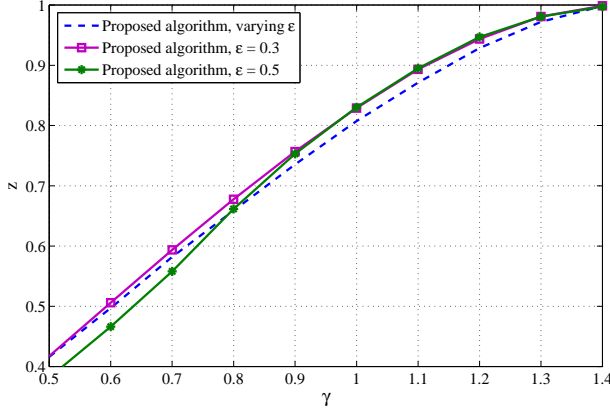


Fig. 5. The intermediate performance of a Raptor code employing our proposed algorithm for the case where ε increases from 0.3 to 0.5 with $\gamma_c = 0.5$.

Figure 5 shows that fluctuation in the ε is well compensated by our algorithm. Since the newly generated c_i 's at γ_c disturb the original sorting order, a slight degradation in z can also be observed. These new c_i 's might have been transmitted earlier than γ_c if they were present in the queue from the beginning. In spite of all the fluctuation in intermediate performance, the code remains capacity achieving since D can collect $k\gamma_{succ}$ encoded symbols in total.

C. Comparison with Fixed-Rate Codes

Since we assumed that an estimate of the channel loss rate, ε , is available at the source, S can employ existing *fixed-rate* codes instead of LT codes to encode \underline{x} . Therefore, we need to compare the performance of our proposed scheme with the intermediate recovery rate of existing fixed-rate codes.

Systematic irregular repeat-accumulate (Systematic IRA) codes [10] are capacity achieving fixed-rate codes on erasure channels, which can provide a high intermediate recovery rate compared to other existing fixed-rate codes. Figure 6 shows the intermediate recovery rate of a systematic IRA code with rate $R = 0.5$ versus our proposed algorithm employing Raptor codes without precoding for two cases of $\varepsilon = 0.44$ and $\varepsilon = 0.46$ with $k = 10000$.

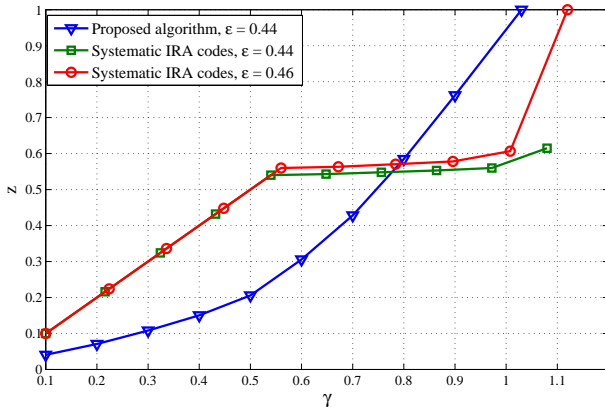


Fig. 6. The intermediate recovery rate of systematic IRA codes [10] compared to our proposed algorithm employing Raptor codes [9] for $k = 10000$.

From Figure 6, we can see that the employed systematic IRA code outperforms our proposed scheme for small values of γ . This high intermediate recovery is due to the systematic part of these codes which

results in transmission of uncoded \underline{x} . When the systematic part of the code is being transmitted, each delivered c_i is itself a source symbol, hence z and γ are equal. However, we can observe that when the transmission of systematic part ends, z does not increase linearly with γ anymore and our proposed scheme outperforms this code for larger values of γ . Further, as $\varepsilon \rightarrow 1$ the systematic part cannot be delivered and the gain from the systematic part is eliminated. In this case, our proposed algorithm always outperforms systematic IRA codes.

Furthermore, fixed-rate codes seriously suffer from their fixed rates since they cannot compensate slight variations in ε . For instance, we can see that when ε is increased from 0.44 to 0.46, z decreases from 1 to 0.6150 at the end of transmission. However, as observed in Figure 5 our algorithm employed along with a Raptor code exhibit good performance in spite of very large variations in ε .

V. CONCLUSION

In this paper, we proposed an algorithm to increase the intermediate recovery rate of capacity achieving LT codes. In our proposed algorithm, the transmitter exploits the structure of LT encoded symbols, a history of previously transmitted encoded symbols, and an estimate of channel's erasure rate to sort the transmission order of the encoded symbols to gain a high intermediate recovery rate.

Using numerical results, we showed that our algorithm can increase the intermediate recovery rate of LT codes to a great extent while the code remains capacity achieving. We also showed that this algorithm performs well for fluctuating channel erasure rates.

To extend this work, we intend to implement the proposed algorithm for real multimedia transmission and observe the improvement made in the quality of the received stream.

REFERENCES

- [1] M. Luby, "LT codes," *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 271–280, 2002.
- [2] S. Sanghavi, "Intermediate performance of rateless codes," *Information Theory Workshop, 2007. ITW '07. IEEE*, pp. 478–482, Sept. 2007.
- [3] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: Maximizing sensor network data persistence," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 255–266, 2006.
- [4] S. Kim and S. Lee, "Improved intermediate performance of rateless codes," in *11th International Conference on Advanced Communication Technology, 2009. ICACT 2009.*, vol. 03, pp. 1682–1686, Feb. 2009.
- [5] A. Talari and N. Rahnavard, "Rateless codes with optimum intermediate performance," *IEEE Globecom 2009 Communication Theory Symposium*, Nov. 2009.
- [6] R. Gummadi and R. Sreenivas, "Relaying a fountain code across multiple nodes," in *IEEE Information Theory Workshop, 2008. ITW '08.*, pp. 149–153, May 2008.
- [7] H. Pishro-Nik, N. Rahnavard, and F. Fekri, "Nonuniform error correction using low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, pp. 2702 – 2714, July 2005.
- [8] D. N. Rowitch and L. B. Milstein, "Rate-compatible punctured turbo (rcpt) codes in a hybrid fec/arq system," in *Proc. Communications Theory Mini-Conf. of IEEE GLOBECOM 97.*, pp. 55–59, Nov 1997.
- [9] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551–2567, June 2006.
- [10] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," pp. 1–8, Sep. 2000.